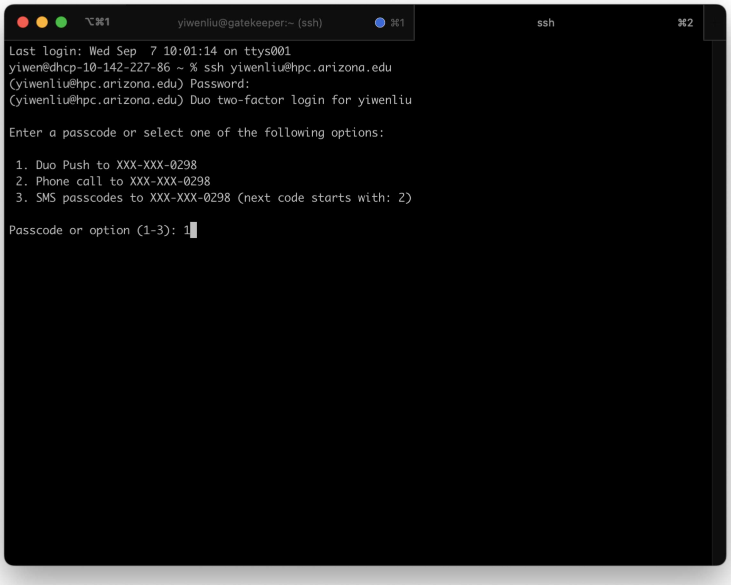
Linux Basics (I)

Healthcare Data Science (BIOS 511)

Teaching server

- On Linux or Mac, access the server by

```
ssh netid@hpc.arizona.edu
```

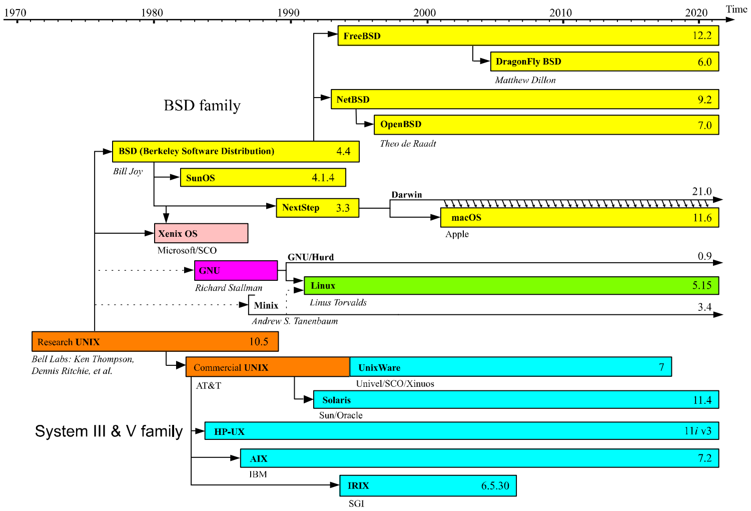


```
Last login: Wed Sep  7 10:01:14 on ttys001
yiwen@dhcp-10-142-227-86 ~ % ssh yiwenliu@hpc.arizona.edu
(yiwenliu@hpc.arizona.edu) Password:
(yiwenliu@hpc.arizona.edu) Duo two-factor login for yiwenliu

Enter a passcode or select one of the following options:

 1. Duo Push to XXX-XXX-0298
 2. Phone call to XXX-XXX-0298
 3. SMS passcodes to XXX-XXX-0298 (next code starts with: 2)

Passcode or option (1-3): 1
```

- Windows machines need the PuTTY (http://www.putty.org) program (free).

What is Linux

Linux is a family of free and open-source software operating systems built around the *Linux kernel*.



Why Linux

Linux is *the* most common platform for scientific computing.

- Open source and community support.
- Things break; when they break using Linux, it's easy to fix.
- Scalability: portable devices (Android, iOS), laptops, servers, clusters, and super computers.
  - E.g. UA HPC cluster, Ocelote and El Gato, runs on Linux.
  - Google cloud computing (GCP) server for this course
- Cost: it's free!

Distributions of Linux (http://upload.wikimedia.org/wikipedia/commons/1/1b/Linux_Distribution_Tim

- Debian/Ubuntu is a popular choice for personal computers.
- RHEL/CentOS is popular on servers.
- The teaching server for this class runs CentOS 7.
- Mac OS was originally derived from Unix/Linux (Darwin kernel). It is POSIX (https://en.wikipedia.org/wiki/POSIX) compliant. Most shell commands we review here apply to Mac OS terminal as well. Windows/DOS, unfortunately, is a totally different breed.
- Show distribution/version on Linux:

  ```
  cat /etc/*-release
  ```

- Show distribution/version on Mac:

  ```
  sw_vers -productVersion
  ```

  ```
  ## 12.5.1
  ```

  or

  ```
  system_profiler SPSoftwareDataType
  ```

Linux shells

What is the shell?

- A shell translates commands to OS instructions. Simply put, the shell is a program that takes commands from the keyboard and gives them to the operating system to perform.
- Most commonly used shells include `bash` , `csh` , `tcsh` , `zsh` , etc.
- Sometimes a script or a command does not run simply because it's written for another shell.
- We mostly use `bash` shell commands in this class.
- Determine the current shell:

```
echo $SHELL
```

```
## /bin/zsh
```

- List available shells:

```
cat /etc/shells
```

```
## # List of acceptable shells for chpass(1).
## # Ftpd will not allow users to connect who are not using
## # one of these shells.
##
## /bin/bash
## /bin/csh
## /bin/dash
## /bin/ksh
## /bin/sh
## /bin/tcsh
## /bin/zsh
```

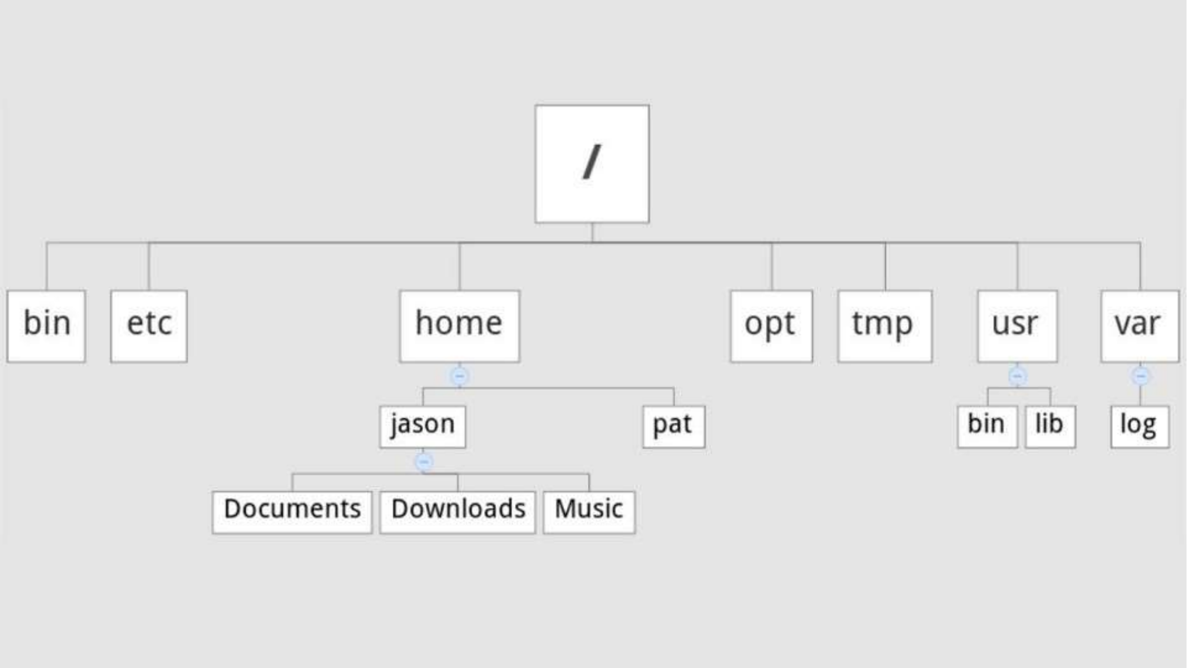
Bash completion

Bash provides the following standard completion for the Linux users by default. Much less typing errors and time!

- Pathname completion.
- Filename completion.
- Variablename completion: `echo $[TAB][TAB]` .
- Username completion: `cd ~[TAB][TAB]` .
- Hostname completion `ssh jzhou@[TAB][TAB]` .
- It can also be customized to auto-complete other stuff such as options and command's arguments. Google `bash completion` for more information.

Navigate file system

Linux directory structure



- /bin – binary or executable programs.
- /etc – system configuration files.
- /home – home directory. It is the default current directory.
- /opt – optional or third-party software.
- /tmp – temporary space, typically cleared on reboot.
- /usr – User related programs.

- /var – log files.
- Upon log in, user is at his/her home directory.

Move around the file system

- `pwd` prints absolute path to the current working directory:

```
pwd
```

```
## /Users/yiwen/Library/CloudStorage/Box-Box/1-MyDocument/Course/BIOS511/lectures/week3/6-Linux basics (I)
```

- `ls` lists contents of a directory:

```
ls
```

```
## Emacs_Reference_Card.pdf
## Example_transcripts.txt
## HowToCreateSSHKeysWithPuTTY.pdf
## Vi_Cheat_Sheet.pdf
## autoSim.R
## exon.txt
## exon_example.txt
## image
## linux.Rmd
## linux.html
## meanEst.R
## meanEst.Rout
## n100.txt
## n200.txt
## n300.txt
## n400.txt
## n500.txt
## output.txt
## runSim.R
## runSim.Rout
## script.R
## script.Rout
```

- `ls -l` lists detailed contents of a directory:

```
ls -l
```

```
## total 4416
## -rw-r--r--@ 1 yiwen  staff   110345 Sep  7 09:58 Emacs_Reference_Card.pdf
## -rw-r--r--@ 1 yiwen  staff      509 Sep  9 13:51 Example_transcripts.txt
## -rw-r--r--@ 1 yiwen  staff   463043 Sep  7 09:58 HowToCreateSSHKeysWithPuTTY.pdf
## -rw-r--r--@ 1 yiwen  staff   200095 Sep  7 09:58 Vi_Cheat_Sheet.pdf
## -rw-r--r--@ 1 yiwen  staff      263 Sep  7 09:58 autoSim.R
## -rw-r--r--@ 1 yiwen  staff      830 Sep 12 09:14 exon.txt
## -rw-r--r--@ 1 yiwen  staff     1300 Sep  9 14:05 exon_example.txt
## drwxr-xr-x  7 yiwen  staff      224 Sep  8 09:34 image
## -rw-r--r--@ 1 yiwen  staff    13182 Sep 12 09:15 linux.Rmd
## -rw-r--r--@ 1 yiwen  staff  1421317 Sep 12 09:14 linux.html
## -rw-r--r--@ 1 yiwen  staff      381 Sep  7 09:58 meanEst.R
## -rw-r--r--@ 1 yiwen  staff     1240 Sep  7 12:35 meanEst.Rout
## -rw-r--r--  1 yiwen  staff        0 Sep  7 10:50 n100.txt
## -rw-r--r--  1 yiwen  staff        0 Sep  7 10:50 n200.txt
## -rw-r--r--  1 yiwen  staff        0 Sep  7 10:50 n300.txt
## -rw-r--r--  1 yiwen  staff        0 Sep  7 10:50 n400.txt
## -rw-r--r--  1 yiwen  staff        0 Sep  7 10:50 n500.txt
## -rw-r--r--@ 1 yiwen  staff        0 Sep  7 12:45 output.txt
## -rw-r--r--@ 1 yiwen  staff      682 Sep  7 09:58 runSim.R
## -rw-r--r--  1 yiwen  staff     1380 Sep  7 10:03 runSim.Rout
## -rw-r--r--@ 1 yiwen  staff      116 Sep  7 12:44 script.R
## -rw-r--r--  1 yiwen  staff      947 Sep  7 12:46 script.Rout
```

- `ls -al` lists all contents of a directory, including those start with `.` (hidden folders):

```
ls -al
```

```
## total 4448
## drwxr-xr-x@ 27 yiwen  staff      864 Sep 12 09:15 .
## drwxr-xr-x   5 yiwen  staff      160 Sep 12 09:15 ..
## -rw-r--r--@  1 yiwen  staff     6148 Sep  7 10:15 .DS_Store
## -rw-r--r--   1 yiwen  staff     3345 Sep  7 12:46 .RData
## -rw-r--r--   1 yiwen  staff      255 Sep  7 16:09 .Rhistory
## -rw-r--r--@  1 yiwen  staff   110345 Sep  7 09:58 Emacs_Reference_Card.pdf
## -rw-r--r--@  1 yiwen  staff      509 Sep  9 13:51 Example_transcripts.txt
## -rw-r--r--@  1 yiwen  staff   463043 Sep  7 09:58 HowToCreateSSHKeysWithPuTTY.pdf
## -rw-r--r--@  1 yiwen  staff   200095 Sep  7 09:58 Vi_Cheat_Sheet.pdf
## -rw-r--r--@  1 yiwen  staff      263 Sep  7 09:58 autoSim.R
## -rw-r--r--@  1 yiwen  staff      830 Sep 12 09:14 exon.txt
## -rw-r--r--@  1 yiwen  staff     1300 Sep  9 14:05 exon_example.txt
## drwxr-xr-x   7 yiwen  staff      224 Sep  8 09:34 image
## -rw-r--r--@  1 yiwen  staff    13182 Sep 12 09:15 linux.Rmd
## -rw-r--r--@  1 yiwen  staff  1421317 Sep 12 09:14 linux.html
## -rw-r--r--@  1 yiwen  staff      381 Sep  7 09:58 meanEst.R
## -rw-r--r--@  1 yiwen  staff     1240 Sep  7 12:35 meanEst.Rout
## -rw-r--r--   1 yiwen  staff        0 Sep  7 10:50 n100.txt
## -rw-r--r--   1 yiwen  staff        0 Sep  7 10:50 n200.txt
## -rw-r--r--   1 yiwen  staff        0 Sep  7 10:50 n300.txt
## -rw-r--r--   1 yiwen  staff        0 Sep  7 10:50 n400.txt
## -rw-r--r--   1 yiwen  staff        0 Sep  7 10:50 n500.txt
## -rw-r--r--@  1 yiwen  staff        0 Sep  7 12:45 output.txt
## -rw-r--r--@  1 yiwen  staff      682 Sep  7 09:58 runSim.R
## -rw-r--r--   1 yiwen  staff     1380 Sep  7 10:03 runSim.Rout
## -rw-r--r--@  1 yiwen  staff      116 Sep  7 12:44 script.R
## -rw-r--r--   1 yiwen  staff      947 Sep  7 12:46 script.Rout
```

- `..` denotes the parent of current working directory.
- `.` denotes the current working directory.
- `~` denotes user’s home directory.
- `/` denotes the root directory.
- `cd ..` changes to parent directory.
- `cd` or `cd ~` changes to home directory.
- `cd /` changes to root directory.

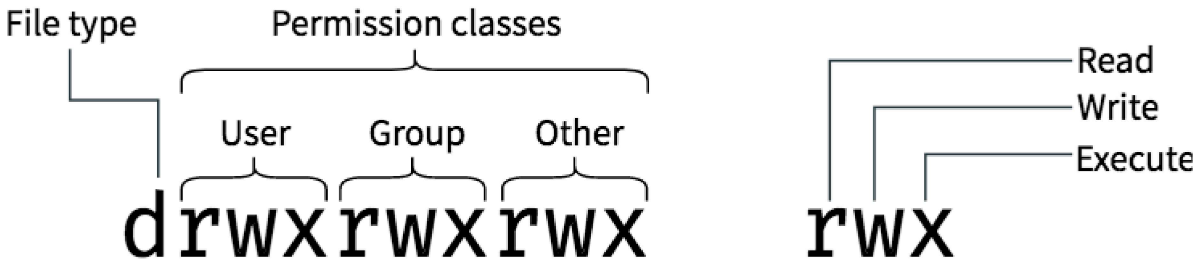
Practices

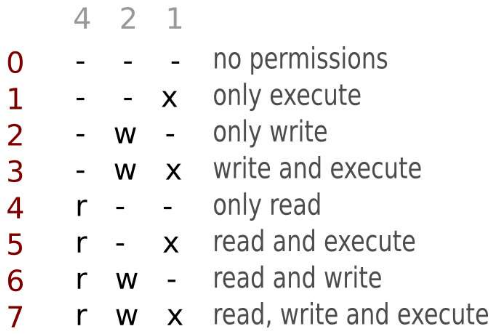
Copy the folder

```
cd /xdisk/yiwenliu
scp -rp ./yiwenliu/linux-basics ./
```

1. Check your current working directory using `pwd`
2. List the content of your home directory using `ls`
3. Go to the `image` folder and list the content of the folder
4. Go the the `miscellaneous` folder and check the content in `random.txt`

File permissions





	4	2	1	
0	-	-	-	no permissions
1	-	-	x	only execute
2	-	w	-	only write
3	-	w	x	write and execute
4	r	-	-	only read
5	r	-	x	read and execute
6	r	w	-	read and write
7	r	w	x	read, write and execute

- `chmod g+x file` makes a file executable to group members.
- `chmod 751 file` sets permission `rwxr-x--x` to a file.
- `groups userid` shows which group(s) a user belongs to:

```
groups yiwenliu
```

Manipulate files and directories

- `cp` copies file to a new location.
- `mv` moves file to a new location.
- `touch` creates a text file; if file already exists, it's left unchanged.
- `rm` deletes a file.
- `mkdir` creates a new directory.
- `rmdir` deletes an *empty* directory.
- `rm -rf` deletes a directory and all contents in that directory (be cautious using the `-f` option …).

Find files

- `which` locates a program:

```
which R
```

```
## /usr/local/bin/R
```

- `find` is similar to `locate` but has more functionalities, e.g., select files by age, size, permissions, …. , and is ubiquitous.

```
find linux.Rmd
```

```
## linux.Rmd
```

```
find -name linux.Rmd
```

Wildcard characters

Wildcard	Matches
`?`	any single character
`*`	any character 0 or more times
`+`	one or more preceding pattern
`^`	beginning of the line
`$`	end of the line
`[set]`	any character in set
`[!set]`	any character not in set
`[a-z]`	any lowercase letter
`[0-9]`	any number (same as `[0123456789]`)

~~~
```bash
# all png files in current folder
ls -l ./image/*.png
```

```
## -rw-r--r--@ 1 yiwen  staff    9507 Sep  7 10:31 ./image/file-permission1.png
## -rw-r--r--@ 1 yiwen  staff   42472 Sep  7 10:30 ./image/file-permission2.png
## -rw-r--r--@ 1 yiwen  staff  433781 Sep  7 10:02 ./image/login.png
```
~~~

Regular expression

- Wildcards are examples of *regular expressions*.
- Regular expressions are a powerful tool to efficiently sift through large amounts of text: record linking, data cleaning, scraping data from website or other data-feed.
- Google `regular expressions` to learn.
- A cheatsheet is available [here][https://cheatography.com/davechild/cheat-sheets/regular-expressions/pdf/ (https://cheatography.com/davechild/cheat-sheets/regular-expressions/pdf/)].

Work with text files

View/peek text files

- `cat` prints the contents of a file:

```
cat linux.Rmd
```

- `head -l` prints the first l lines of a file:

```
head linux.Rmd
```

```
## ---
## title: "Linux Basics (I)"
## author:
## date:
## output:
##   html_document:
##     toc: yes
## subtitle: Healthcare Data Science (BIOS 511)
## ---
```

- `tail -l` prints the last l lines of a file:

```
tail linux.Rmd
```

```
##     - `:wq<Return>` quits `vi` and saves changes.
##
## - Google `vi cheatsheet`
##
## ### IDE (Integrated Development Environment)
##
## - Statisticians write a lot of code. Critical to adopt a good IDE that goes beyond code editing: syntax highlighting, executing code within editor, debugging, profiling, version control, etc.
##
## - R Studio, Eclipse, Emacs, Matlab, Visual Studio, etc.
```

- `less` is more; `more` is less

`more` browses a text file screen by screen (only downwards). Scroll down one page (paging) by pressing the spacebar; exit by pressing the `q` key. `less` is also a pager, but has more functionalities, e.g., scroll upwards and downwards through the input. `less` doesn't need to read the whole file, i.e., it loads files faster than `more` .

Piping and redirection

- `|` sends output from one command as input of another command.
- `>` directs output from one command to a file.
- `>>` appends output from one command to a file.
- `<` reads input from a file.

Common operations

`grep`

`grep` prints lines that match an expression:

- Show lines that contain string `CentOS` :

```
# quotes not necessary if not a regular expression
grep 'CentOS' linux.Rmd
```

```
## - RHEL/CentOS is popular on servers.
## - The teaching server for this class runs CentOS 7.
## - Show lines that contain string `CentOS`:
##     grep 'CentOS' linux.Rmd
##     grep 'CentOS' *.Rmd
##     grep -n 'CentOS' linux.Rmd
## - Replace `CentOS` by `RHEL` in a text file:
##     sed 's/CentOS/RHEL/' linux.Rmd | grep RHEL
```

- Search multiple text files:

```
grep 'CentOS' *.Rmd
```

```
## - RHEL/CentOS is popular on servers.
## - The teaching server for this class runs CentOS 7.
## - Show lines that contain string `CentOS`:
##     grep 'CentOS' linux.Rmd
##     grep 'CentOS' *.Rmd
##     grep -n 'CentOS' linux.Rmd
## - Replace `CentOS` by `RHEL` in a text file:
##     sed 's/CentOS/RHEL/' linux.Rmd | grep RHEL
```

- Show matching line numbers:

```
grep -n 'CentOS' linux.Rmd
```

```
## 49:- RHEL/CentOS is popular on servers.
## 51:- The teaching server for this class runs CentOS 7.
## 321:- Show lines that contain string `CentOS`:
## 324:    grep 'CentOS' linux.Rmd
## 329:    grep 'CentOS' *.Rmd
## 334:    grep -n 'CentOS' linux.Rmd
## 369:- Replace `CentOS` by `RHEL` in a text file:
## 371:    sed 's/CentOS/RHEL/' linux.Rmd | grep RHEL
```

- Find all files in current directory with `.png` extension:

```
cd ./image
ls | grep '\.png$'
```

```
## file-permission1.png
## file-permission2.png
## login.png
```

- Find all directories in the current directory:

```
ls -al | grep '^d'
```

```
## drwxr-xr-x@ 27 yiwen  staff      864 Sep 12 09:15 .
## drwxr-xr-x   5 yiwen  staff      160 Sep 12 09:15 ..
## drwxr-xr-x   7 yiwen  staff      224 Sep  8 09:34 image
```

- Practice

1. check the content in the file `mysampledata.txt` in the folder `miscellaneous` .

```
cat mysampledata.txt
```

2. identify every line which contained the string `mellon` .

```
grep -n 'mellon' mysampledata.txt
```

3. identify everyone who’s name begins with A - K.

```
grep '^[A-K]' mysampledata.txt
```

sed

- `sed` is a stream editor.
- Replace `CentOS` by `RHEL` in a text file:

```
sed 's/CentOS/RHEL/' linux.Rmd | grep RHEL
```

```
## - RHEL/RHEL is popular on servers.
## - The teaching server for this class runs RHEL 7.
## - Show lines that contain string `RHEL`:
##     grep 'RHEL' linux.Rmd
##     grep 'RHEL' *.Rmd
##     grep -n 'RHEL' linux.Rmd
## - Replace `RHEL` by `RHEL` in a text file:
##     sed 's/RHEL/RHEL/' linux.Rmd | grep RHEL
```

- ‘s’ specifies the substitution operation

awk

`awk` is a filter and report writer with syntax: `awk <command> infile.txt > outfile.txt`

- Print the first column of `Example_transcript.txt`

```
awk '{print $1}' Example_transcripts.txt
```

```
## Transcript_ID
## T_0001
## T_0002
## T_0003
## T_0004
## T_0005
## T_0006
## T_0007
## T_0008
## T_0009
## T_0010
## T_0011
## T_0012
## T_0013
## T_0014
## T_0015
```

- Print the columns 1, 3, and 5 of `Example_transcript.txt`

```
awk '{print $1, $3, $5}' Example_transcripts.txt
```

```
## Transcript_ID Untreated_abundance Change
## T_0001 200 Down
## T_0002 50 Down
## T_0003 50 Up
## T_0004 250 No_change
## T_0005 50 No_change
## T_0006 25 No_change
## T_0007 100 No_change
## T_0008 500 No_change
## T_0009 25 Up
## T_0010 100 No_change
## T_0011 300 No_change
## T_0012 100 No_change
## T_0013 100 Up
## T_0014 50 Up
## T_0015 125 No_change
```

The real power of AWK is in its ability to filter files for specific values in specified columns. A GTF file typically contains many different genomic features. Here is a GTF-like file to play with.

```
head -5 exon_example.txt
```

```
## 1    gene    1000    2000    "gene_id ""GOI1""; exon_number ""3"";"
## 1    transcript  1000    2000    "gene_id ""GOI1""; transcript_id ""GOI1.1""; exon_number ""3"";"
## 1    transcript  1000    2000    "gene_id ""GOI1""; transcript_id ""GOI1.2""; exon_number ""2"";"
## 1    exon    1000    1300    "gene_id ""GOI1""; transcript_id ""GOI1.1""; exon_number ""1"";"
## 1    exon    1400    1500    "gene_id ""GOI1""; transcript_id ""GOI1.1""; exon_number ""2"";"
```

- If we are only interested in the exon features, select for the presence of `exon` in column 2.

```
awk ' $2=="exon" ' exon_example.txt
```

```
## 1    exon    1000    1300    "gene_id ""GOI1""; transcript_id ""GOI1.1""; exon_number ""1"";"
## 1    exon    1400    1500    "gene_id ""GOI1""; transcript_id ""GOI1.1""; exon_number ""2"";"
## 1    exon    1600    2000    "gene_id ""GOI1""; transcript_id ""GOI1.1""; exon_number ""3"";"
## 1    exon    1000    1300    "gene_id ""GOI1""; transcript_id ""GOI1.2""; exon_number ""1"";"
## 1    exon    1600    2000    "gene_id ""GOI1""; transcript_id ""GOI1.2""; exon_number ""2"";"
## 1    exon    5000    5500    "gene_id ""GOI2""; transcript_id ""GOI2.1""; exon_number ""1"";"
## 1    exon    5600    5900    "gene_id ""GOI2""; transcript_id ""GOI2.1""; exon_number ""2"";"
## 1    exon    6000    7000    "gene_id ""GOI2""; transcript_id ""GOI2.1""; exon_number ""3"";"
## 1    exon    5000    5500    "gene_id ""GOI2""; transcript_id ""GOI2.2""; exon_number ""1"";"
## 1    exon    6000    6500    "gene_id ""GOI2""; transcript_id ""GOI2.2""; exon_number ""2"";"
```

`awk` can also filter numerical values. No need to convert the number values from strings to integers/floats, `awk` does that for you!

- Get values greater or equal to 100 in column 4 of file `Example_transcripts.txt` (notice that there are no quotation marks (“”) around the value to be filtered as with text values).

```
awk '$4 >= 100' Example_transcripts.txt
```

```
## Transcript_ID    Gene_name   Untreated_abundance Treated_abundance   Change
## T_0003   RS2Z37  50  150 Up
## T_0004   RS2Z37  250 250 No_change
## T_0007   TOPLESS 100 100 No_change
## T_0008   EF1alpha    500 500 No_change
## T_0009   RS2Z38  25  100 Up
## T_0010   RS2Z38  100 100 No_change
## T_0011   ANR 300 300 No_change
## T_0012   PEX5    100 100 No_change
## T_0013   eIF5L1  100 250 Up
## T_0014   SMG7-2  50  200 Up
## T_0015   LUG 125 125 No_change
```

- A couple more conditions

```
awk '$3 > 100 && $4 > 100' Example_transcripts.txt
```

```
## Transcript_ID    Gene_name   Untreated_abundance Treated_abundance   Change
## T_0004   RS2Z37  250 250 No_change
## T_0008   EF1alpha    500 500 No_change
## T_0011   ANR 300 300 No_change
## T_0015   LUG 125 125 No_change
```

- A couple more examples

```
awk '$3 > 100 && $4 > 100 {print $1, $3}' Example_transcripts.txt
```

```
## Transcript_ID Untreated_abundance
## T_0004 250
## T_0008 500
## T_0011 300
## T_0015 125
```

`awk` uses a special rule called `END` . NR represents number of rows, and NF represents number of fields or variables.

```
  awk 'END {print NR}' Example_transcripts.txt
  awk 'END {print NF}' Example_transcripts.txt
```

```
## 16
## 5
```

OR

```
  awk ' BEGIN {i=0}{i++;} END {print i} ' Example_transcripts.txt
```

```
## 16
```

Piping and redirection

Combinations of shell commands (`grep` , `sed` , `awk` , …), piping and redirection, and regular expressions allow us pre-process and reformat huge text files efficiently.

- select for the presence of `exon` in column 2 and save the data as `exon.txt`

```
awk ' $2=="exon" ' exon_example.txt > exon.txt
```

- Print the subject name in `mysampledata.txt` , sort the output according to alphabetic order, and output the sorted names to a file named `sortdata.txt`

```
awk '{print $1}' mysampledata.txt | sort > sortdata.txt
```

Text editors

Emacs

- `Emacs` is a powerful text editor with extensive support for many languages including `R` , $\LaTeX$, `python` , and `C/C++` ; however it's *not* installed by default on many Linux distributions.

- Basic survival commands:
  - `emacs filename` to open a file with emacs.
  - `CTRL-x CTRL-f` to open an existing or new file.
  - `CTRL-x CTRX-s` to save.
  - `CTRL-x CTRL-w` to save as.
  - `CTRL-x CTRL-c` to quit.
- Google `emacs cheatsheet`

`C-<key>` means hold the `control` key, and press `<key>` .
`M-<key>` means press the `Esc` key once, and press `<key>` .

Vi

- `Vi` is ubiquitous (POSIX standard). Learn at least its basics; otherwise you can edit nothing on some clusters.

- Basic survival commands:
  - `vi filename` to start editing a file.
  - `vi` is a *modal* editor: *insert* mode and *normal* mode. Pressing `i` switches from the normal mode to insert mode. Pressing `ESC` switches from the insert mode to normal mode.
  - `:x<Return>` quits `vi` and saves changes.
  - `:q!<Return>` quits vi without saving latest changes.
  - `:w<Return>` saves changes.
  - `:wq<Return>` quits `vi` and saves changes.
- Google `vi cheatsheet`

IDE (Integrated Development Environment)

- Statisticians write a lot of code. Critical to adopt a good IDE that goes beyond code editing: syntax highlighting, executing code within editor, debugging, profiling, version control, etc.

- R Studio, Eclipse, Emacs, Matlab, Visual Studio, etc.